Quicksort Correctness Proof

https://cs.pomona.edu/classes/cs140/
Outline

Topics and Learning Objectives
• Learn how quicksort works
• Learn how to partition an array

Exercise
• Loop Invariant
Extra Resources

- https://me.dt.in.th/page/Quicksort/
- https://www.youtube.com/watch?v=ywWBy6J5gz8
- CLRS Chapter 7
What do we need to do?

1. Prove that Partition works
2. Prove that QuickSort works
| 31 | 47 | 11 | 91 | 67 | 23 | 89 | 51 |
Not a copy! (In-place)

(partition)

(doesn't match function from slides)
Not a copy!

(partition)

(base)

(base)

(doesn't match function from slides)
Not a copy!

11 23 31 51 89 67 91 47

Base

Not a copy!

11 23 31 51 89 67 91 47

(partition)

doesn't match function from slides

Base

11 23 31 51 89 67 91 47
Not a copy!

11 23 31 51 89 67 91 47

(partition)

Not a copy!

11 23 31 51 89 67 91 47

(base)

11 23 31 51 89 67 91 47

(base)

11 23 31 47 51 91 67 89

(base)
### Partition proof of correctness

<table>
<thead>
<tr>
<th>Value</th>
<th>67</th>
<th>44</th>
<th>...</th>
<th>21</th>
<th>-87</th>
<th>...</th>
<th>5</th>
<th>101</th>
<th>-31</th>
<th>...</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>left</td>
<td>left + 1</td>
<td>...</td>
<td>right - 1</td>
<td>right</td>
<td>right + 1</td>
<td>...</td>
<td>n - 1</td>
</tr>
</tbody>
</table>

**FUNCTION** `Partition(array, left_index, right_index)`

1. `pivot_value = array[left_index]`
2. `i = left_index + 1`
3. **FOR** `j IN [left_index + 1 ..< right_index]`
   - **IF** `array[j] < pivot_value`
     - `swap(array, i, j)`
     - `i = i + 1`
4. `swap(array, left_index, i - 1)`
5. **RETURN** `i - 1`
Function \texttt{Partition}(array, left\_index, right\_index)

\begin{align*}
\text{pivot\_value} &= \text{array}[\text{left\_index}] \\
i &= \text{left\_index} + 1 \\
\text{FOR } j \text{ IN } [\text{left\_index} + 1 .. < \text{right\_index}] & \\
\text{IF } \text{array}[j] < \text{pivot\_value} & \\
\text{IF } \text{array}[j] < \text{pivot\_value} & \\
\text{swap}(\text{array}, i, j) & \\
i &= i + 1 \\
\text{swap}(\text{array}, \text{left\_index}, i - 1) & \\
\text{RETURN } i - 1
\end{align*}
Loop Invariant Proofs

1. State the loop invariant
   1. A statement that can be easily proven true or false
   2. The statement should reference the purpose of the loop
   3. The statement should reference variables that change each iteration

2. Show that the loop invariant is true before the loop starts

3. Show that the loop invariant holds when executing any iteration

4. Show that the loop invariant holds once the loop ends
Loop Invariant

• Loop invariants will usually look something like this:

At the start of the iteration with index i, ... the subarray array[0 .. ] ...
FUNCTION Partition(array, left_index, right_index)

pivot_value = array[left_index]
i = left_index + 1

FOR j IN [left_index + 1 ..< right_index]
   IF array[j] < pivot_value
      swap(array, i, j)
i = i + 1

swap(array, left_index, i - 1)

RETURN i - 1
Partition proof of correctness

<table>
<thead>
<tr>
<th>Value</th>
<th>67</th>
<th>44</th>
<th>...</th>
<th>21</th>
<th>-87</th>
<th>...</th>
<th>5</th>
<th>101</th>
<th>-31</th>
<th>...</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>left</td>
<td>left + 1</td>
<td>...</td>
<td>right - 1</td>
<td>right</td>
<td>right + 1</td>
<td>...</td>
<td>n - 1</td>
</tr>
</tbody>
</table>

**FUNCTION** Partition(array, left_index, right_index)

pivot_value = array[left_index]

i = left_index + 1

FOR j IN [left_index + 1 ..< right_index]

IF array[j] < pivot_value

    swap(array, i, j)

    i = i + 1

swap(array, left_index, i - 1)

RETURN i - 1
Partition proof of correctness

<table>
<thead>
<tr>
<th>Value</th>
<th>67</th>
<th>44</th>
<th>…</th>
<th>21</th>
<th>-87</th>
<th>…</th>
<th>5</th>
<th>101</th>
<th>-31</th>
<th>…</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>0</td>
<td>1</td>
<td>…</td>
<td>left</td>
<td>left + 1</td>
<td>…</td>
<td>right - 1</td>
<td>right</td>
<td>right + 1</td>
<td>…</td>
<td>n - 1</td>
</tr>
</tbody>
</table>

**FUNCTION Partition** (array, left_index, right_index)

pivot_value = array[left_index]

i = left_index + 1

FOR j IN [left_index + 1 ..< right_index]

IF array[j] < pivot_value

swap(array, i, j)

i = i + 1

swap(array, left_index, i - 1)

RETURN i - 1

Loop Invariant: At the start of the iteration with indices i and j:
1. All items in array[l+1 ..= i-1] are < pivot_value
2. All items in array[i ..= j-1] are ≥ pivot_value

How do we prove that Partition is correct?
**Loop Invariant:** At the start of the iteration with indices i and j:
1. All items in \( a[l+1 ..= i-1] \) are \(<\) pivot
2. All items in \( a[i ..= j-1] \) are \(\geq\) pivot

**FUNCTION** Partition(a, l, r)

\[
\begin{align*}
\text{pivot\_value} & = a[l] \\
n & = l + 1 \\
\text{FOR } j \text{ IN } [l + 1 ..< r] & \\
\text{IF } a[j] < \text{pivot\_value} & \\
\text{swap}(a, i, j) & \\
\end{align*}
\]

\[
\begin{align*}
i & = i + 1 \\
\text{swap}(a, l, i - 1) & \\
\text{RETURN } i - 1 & \\
\end{align*}
\]
**Partition Proof**

**Initialization:** Show that the loop invariant is true before the loop starts

1. No numbers in $a[l+1 ..= i-1]$
2. No numbers in $a[i ..= j-1]$

**Loop Invariant:** At the start of the iteration with indices $i$ and $j$:

1. All items in $a[l+1 ..= i-1]$ are $< \text{pivot}$
2. All items in $a[i ..= j-1]$ are $\geq \text{pivot}$

**FUNCTION**  

```plaintext
Partition(a, l, r)  
pivot_value = a[l]  
i = l + 1  
FOR j IN [l + 1 ..< r]  
  IF a[j] < pivot_value  
    swap(a, i, j)  
    i = i + 1  
  swap(a, l, i - 1)  
RETURN i - 1
```
**Partition Proof**

**Function**

\[
\text{Partition}(a, l, r) = \\
\text{pivot}\_\text{value} = a[l] \\
i = l + 1 \\
\text{FOR } j \text{ IN } [l + 1 .. < r] \\
\text{IF } a[j] < \text{pivot}\_\text{value} \\
\text{swap}(a, i, j) \\
i = i + 1 \\
\text{swap}(a, l, i - 1) \\
\text{RETURN } i - 1
\]

**Loop Invariant:** At the start of the iteration with indices i and j:
1. All items in \(a[l+1 ..= i-1]\) are \(< \text{pivot}\)
2. All items in \(a[i ..= j-1]\) are \(\geq \text{pivot}\)

**Maintenance (case 1):** Show that the loop invariant holds when executing any iteration

- Suppose conditions 1 and 2 are met.
- Now, suppose \(a[j] < \text{pivot}\)
Partition Proof

Maintenance (case 1):
- Suppose conditions 1 and 2 are met.
- Now, suppose $a[j] < \text{pivot}$
- Then $a[j]$ and $a[i]$ are swapped
- By (2), $a[i]$ was $> \text{pivot}$ so now $a[i] < \text{pivot}$ and $a[j] > \text{pivot}$

Loop Invariant: At the start of the iteration with indices $i$ and $j$:
1. All items in $a[l+1 ..= i-1]$ are $< \text{pivot}$
2. All items in $a[i ..= j-1]$ are $\geq \text{pivot}$

FUNCTION Partition($a, l, r$)

\[
\text{pivot\_value} = a[l]
\]
\[
i = l + 1
\]
\[
\text{FOR } j \text{ IN } [l + 1 ..< r]
\]
\[
\text{IF } a[j] < \text{pivot\_value}
\]
\[
\text{swap}(a, i, j)
\]
\[
i = i + 1
\]
\[
\text{swap}(a, l, i - 1)
\]
\[
\text{RETURN } i - 1
\]
Partition Proof

FUNCTION Partition(a, l, r)
pivot_value = a[l]
i = l + 1
FOR j IN [l + 1 ..< r]
    IF a[j] < pivot_value
        swap(a, i, j)
i = i + 1
    swap(a, l, i - 1)
RETURN i - 1

Loop Invariant: At the start of the iteration with indices i and j:
1. All items in a[l+1 ..= i-1] are < pivot
2. All items in a[i ..= j-1] are ≥ pivot

Maintenance (case 1):
• Suppose conditions 1 and 2 are met.
• Now, suppose a[j] < pivot
• Then a[j] and a[i] are swapped
• By (2), a[i] was > pivot so now a[i] < pivot and a[j] > pivot
**Partition Proof**

**FUNCTION** Partition(a, l, r)

\[
\text{pivot\_value} = a[l]
\]

\[i = l + 1\]

\[\text{FOR } j \text{ IN } [l + 1 ..< r]\]

\[\text{IF } a[j] < \text{pivot\_value}\]

\[\text{swap}(a, i, j)\]

\[i = i + 1\]

\[\text{swap}(a, l, i - 1)\]

\[\text{RETURN } i - 1\]

**Loop Invariant:** At the start of the iteration with indices i and j:
1. All items in \(a[l+1 ..= i-1]\) are \(<\) pivot
2. All items in \(a[i ..= j-1]\) are \(\geq\) pivot

**Maintenance (case 1):**
- Suppose conditions 1 and 2 are met.
- Now, suppose \(a[j] < \text{pivot}\)
- Then \(a[j]\) and \(a[i]\) are swapped
- By (2), \(a[i]\) was \(>\) pivot so now \(a[i] < \text{pivot}\) and \(a[j] > \text{pivot}\)
- Incrementing \(i\) and \(j\) satisfies 1 and 2
**FUNCTION** Partition(a, l, r)

```plaintext
pivot_value = a[l]
i = l + 1

FOR j IN [l + 1 ..< r]
IF a[j] < pivot_value
    swap(a, i, j)
i = i + 1

swap(a, l, i - 1)
RETURN i - 1
```

**Loop Invariant:** At the start of the iteration with indices i and j:
1. All items in a[l+1 ..= i-1] are < pivot
2. All items in a[i ..= j-1] are ≥ pivot

**Maintenance (case 2):**
- Suppose conditions 1 and 2 are met.
- Now, suppose a[j] ≥ pivot
FUNCTION Partition(a, l, r)
    pivot_value = a[l]
    i = l + 1
    FOR j IN [l + 1 ..< r]
        IF a[j] < pivot_value
            swap(a, i, j)
            i = i + 1
        swap(a, l, i - 1)
    RETURN i - 1

Loop Invariant: At the start of the iteration with indices i and j:
1. All items in a[l+1 ..= i-1] are < pivot
2. All items in a[i ..= j-1] are ≥ pivot

Maintenance (case 2):
- Suppose conditions 1 and 2 are met.
- Now, suppose a[j] ≥ pivot
- We do not change i so (1) holds
FUNCTION Partition(a, l, r)
    pivot_value = a[l]
    i = l + 1
    FOR j IN [l + 1 ..< r]
        IF a[j] < pivot_value
            swap(a, i, j)
            i = i + 1
    swap(a, l, i - 1)
    RETURN i - 1

Loop Invariant: At the start of the iteration with indices i and j:
1. All items in a[l+1 ..= i-1] are < pivot
2. All items in a[i ..= j-1] are ≥ pivot

Maintenance (case 2):
• Suppose conditions 1 and 2 are met.
• Now, suppose a[j] ≥ pivot
• We do not change i so (1) holds
• We increment j so (2) holds
**Termination**: Show that the loop invariant holds once the loop ends

- Now \( j = r \)
- All items have been considered
- All items in \( a[l+1 \ldots i-1] \) are < \( \text{pivot} \)
- All items in \( a[i \ldots j-1] \) are ≥ \( \text{pivot} \)

**Loop Invariant**: At the start of the iteration with indices \( i \) and \( j \):
1. All items in \( a[l+1 \ldots i-1] \) are < \( \text{pivot} \)
2. All items in \( a[i \ldots j-1] \) are ≥ \( \text{pivot} \)

**FUNCTION** Partition\((a, l, r)\)

\[
\begin{align*}
\text{pivot\_value} &= a[l] \\
i &= l + 1 \\
\text{FOR } j \text{ IN } [l + 1 \ldots < r] \\
&\quad \text{IF } a[j] < \text{pivot\_value} \\
&\quad \quad \text{swap}(a, i, j) \\
&\quad \quad i = i + 1 \\
&\quad \text{swap}(a, l, i - 1) \\
&\quad \text{RETURN } i - 1
\end{align*}
\]
**Partition Proof**

**FUNCTION** Partition(a, l, r)

pivot_value = a[l]

i = l + 1

FOR j IN [l + 1 ..< r]

IF a[j] < pivot_value
    swap(a, i, j)
    i = i + 1

swap(a, l, i - 1)

RETURN i - 1

**Loop Invariant:** At the start of the iteration with indices i and j:
1. All items in a[l+1 ..= i-1] are < pivot
2. All items in a[i ..= j-1] are ≥ pivot

After the loop we do the final swap
What do we need to do?

1. Prove that PARTITION works
   • Proof by loop invariant

2. Prove that Quicksort works
   • Proof by induction
Proof by Induction in General

Some property $P$ that we want to prove

• A **base case**: some statement regarding $P(1)$
• An **inductive hypothesis**: assume we know that $P(n)$ is true
• An **inductive step**: if $P(n)$ is correct then so is $P(n+1)$ because...

For quicksort we are going to use a slightly different form

• If $P(k)$ where $k < n$ is correct, then $P(n)$ is also correct
• An **inductive hypothesis**: assume we know that $P(k)$ is true
• An **inductive step**: if $P(k)$ is correct then so is $P(n)$ because...
Proof by Induction Cheat-sheet

Proof by induction that $P(n)$ holds for all $n$
1. $P(1)$ holds because <something about the code/problem>
2. Let’s assume that $P(k)$ (where $k < n$) holds.
3. $P(n)$ holds because of $P(k)$ and <something about the code>
4. Thus, by induction, $P(n)$ holds for all $n$
Quicksort Proof

P(n) = Quicksort is always correct for arrays of length n.
• P(1) is an array of one element, and any such array is always sorted.
• Assume (hypothesis) that P(k) is correct for k < n
• P(n) holds because:
Quicksort Proof

\[ P(n) = \text{Quicksort is always correct for arrays of length } n. \]

- \( P(1) \) is an array of one element, and any such array is always sorted.
- **Assume (hypothesis)** that \( P(k) \) is correct for \( k < n \)
- \( P(n) \) holds because:
  - Let \( k_{\text{left}}, k_{\text{right}} \) = the lengths of the left and right subarrays
  - \( k_{\text{left}}, k_{\text{right}} < n \) (strictly less than \( n \))
  - By our *inductive hypothesis*, the left and right subarrays are correctly sorted
  - The partition loop-invariant guarantees that the pivot is in the correct spot

Proof by induction that \( P(n) \) holds for all \( n \)
- \( P(1) \) holds because ...
- Let’s assume that \( P(k) \) (where \( k < n \)) holds.
- \( P(n) \) holds because of \( P(k) \) and ...
- Thus, by induction, \( P(n) \) holds for all \( n \)
**Quicksort Proof**

\[ P(n) = \text{Quicksort is always correct for arrays of length } n. \]

- **P(1)** is an array of one element, and any such array is always sorted. *(Base case)*

- Assume *(hypothesis)* that \( P(k) \) is correct for \( k < n \). *(Inductive Hypothesis)*

- \( P(n) \) holds because:
  - Let \( k_{\text{left}}, k_{\text{right}} \) = the lengths of the left and right subarrays
  - \( k_{\text{left}}, k_{\text{right}} < n \) (strictly less than \( n \))
  - By our hypothesis, the left and right subarrays are correctly sorted
  - The partition loop-invariant guarantees that the pivot is in the correct spot

---

Proof by induction that \( P(n) \) holds for all \( n \)

- \( P(1) \) holds because ...
- Let’s assume that \( P(k) \) (where \( k < n \)) holds.
- \( P(n) \) holds because of \( P(k) \) and ...
- Thus, by induction, \( P(n) \) holds for all \( n \)
What do we need to do?

1. Prove that PARTITION works
   • Proof by loop invariant

2. Prove that Quicksort works
   • Proof by induction